基于素数编码技术对供应链中路径编码的研究

陈 雄, 王笑梅

(上海师范大学 信息与机电工程学院, 上海 200234)

摘 要: 无线射频识别(RFID)技术已经应用到许多不同的领域,尤其在供应链的监测和管理中是非常有用的,然而在这样的环境中将产生巨大的路径信息数据,从这些数据中提取有用的信息需要耗费很长时间. 采用目前广泛使用的编码技术素数编码对供应链中物品路径进行有效编码,减小存储空间,并方便地检索路径信息. 设计了一个存储策略支持在关系型数据库上进行有效的查询处理. 最后提出了一种将查询模板转换为SQL方法.

关键词: 射频识别; 供应链; 素数编码; 存储策略; 查询模板

中图分类号: TP 391 **文献标识码:** A **文章编号:** 1000-5137(2012)05-0483-07

0 引 言

供应链是通过对信息流、物流、资金流的控制,实现将供应商、制造商、分销商、零售商直到最终用户连成一个整体的功能结构网络. 供应链系统的一个重要功能就是能对物品进行跟踪和追溯. 当发现质量问题时,能根据这些流通记录追溯问题商品的批次、责任企业和个人[1]. 因此对供应链物品的流通路径进行追溯显得日益重要.

将射频识别技术[2](Radio Frequent Identification,RFID)引入到物流与供应链管理中,使管理效率得到了极大的提高. 现代物流中,存在有大量的移动物品,RFID技术可以自动记录这些物品的移动痕迹,产生海量的路径信息. 通过对海量路径信息的有效处理,用户可以全程监控物品的流动情况,提高库存的流动速度,减少管理人员,降低成本.

近年来,人们提出许多基于树的编码方法来处理路径信息,即按照某种规则对树中每个结点分配唯一编码. 其目的是通过编码值就能判断任意两个结点之间是否具有祖先-后代等结构关系,以支持高效查询. 目前,广泛使用的主要编码方法包括区间编码、前缀编码、向量编码及素数编码等.

基于区间的编码方法[3]用区间来表示结点的编码,支持对XML文档的查询,但不支持节点的任意插入. 即使之前为新插入结点预留了一部分编码空间,在有新的结点插入时不得不对文档中所有结点进行重新编码. 另一种编码方法是前缀编码方式[4],这种方式通过结点编码前缀来确定结点间的关系,对于前缀编码方法,当在某一个位置插入一个新的结点时,它影响该插入结点的父节点的所有子孙结点. 前缀编码方式同全序编码方式[5]一样,能唯一地确定结点的关系,有效地支持文档的查询和更新,在有新的结点插入时不会影响其他已存在结点的编码,但存储结点编码路径需要额外的空间. 向量编码(vector encoding)[6,7]是一种垂直的编码方法. 它的基本思想是: 用向量代替整数,将在2个整数之间插入一个整数转化为在2个向量之间插入一个向量. 因为在向量间可以插入无限个向量,因此可以避免重新编码问题,解决更新问题. 但其向量的计算依赖于区间编码,且不能直接反映出结点是否是叶子节点

收稿日期: 2012-09-25

作者简介: 陈 雄(1986-),男,上海师范大学信息与机电工程学院硕士研究生; 王笑梅(1970-),女,上海师范大学信息与机电工程学院副教授.

以及包含关系动态变化的历史. 素数编码方式[8-11]是利用素数的唯一性为结点编码. 这种编码保证在同一文档树中,祖先结点的编码一定能够整除后代结点编码,能有效地支持文档的更新和查询,其中由于素数具有唯一性和独立性,所以即使图中某一结点发生变化,也不会影响其他结点的编码,很大程度减小了更新代价,而且任意结点只要查询一次即可知道所有的祖先关系,查询效率高,但这种编码方式随着结点的插入,结点的编码会越来越大.

本文作者基于素数编码技术对供应链中的路径进行有效的编码,减小存储空间. 主要解决出现重复路径及路径过长导致节点编码过大的问题,并基于该编码策略设计一个关系模型来存储路径编码后的信息,并将定义的查询模板转化为 SQL 语句支持的有效处理物品追踪查询和路径查询.

1 结构模型

结构模型如图 1,从 RFID 原始数据中提取的路径信息经过编码后用元素列表编码和顺序编码存储在关系数据库中,由于用素数代替了位置名字,所以(位置、素数)以哈希结构保存在内存中,编码后的路径信息存储在 PATH_TABLE 表中,产品的详细信息放在 INFO_TABLE 表中,IN_TABLE 为关联表. 如果用户追踪查询,查询将被转化为 SQL 查询,查询结果将返回给用户.

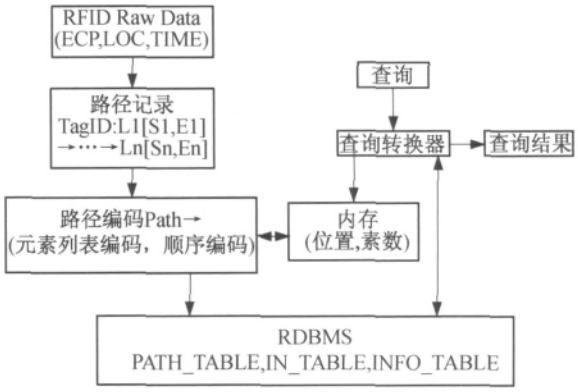

图 1 结构模型

由 RFID 阅读器进行信息的读取,得到形如元组(EPC,位置,时间)的 RFID 记录,其中"位置"表示阅读器所在的位置,"时间"表示读取的时间. 为了便于说明,现给出清理前的 RFID 数据如图 2a 所示,使用 RFID 数据清理得到形如元组(EPC,位置,进入时间,离开时间)的 stay 记录如图 2b 所示,其中"进入时间"表示物品进入阅读器作用范围的时间,"离开时间"表示物品离开阅读器作用范围的时间. 路径(path)如图 2c 所示由一系列具有相同 EPC 形如(位置,进入时间,离开时间)的元组按照一定的顺序排列组成,如 $x=(d1,s1,e1)(d2,s2,e2)(d3,s2,e3)$ 就是一条路径.

(tag1,A,2),(tag1,A,3)	Tag1:A[2,3]	Tag1:A[2,3]→B[5,7]→C[8,9]
(tag2,A,2),(tag2,A,3)	Tag1:B[5,7]	Tag2:A[2,3]→B[5,7]→D[8,9]
(tag1,B,5),(tag1,B,7)	Tag1:C[8,9]	
(tag2,B,5),(tag2,B,7)	Tag2:A[2,3]	
(tag1,C,8),(tag1,C,9)	Tag2:B[5,7]	
(tag2,D,8),(tag2,D,9)	Tag2:D[5,7]	
(a)RFID记录	(b)Stay记录	(c)路径

图 2 数据记录

2 路径处理

2.1 素数编码

这里首先介绍两个著名的定理[12]算术基本定理(唯一分解定理)和中国剩余定理.

定理 1 算术基本定理(唯一分解定理):任何超过 1 的自然数都可以用一些唯一的素数的乘积表示.

例如,$30=235$,任何其他素数相乘等于 30 都不存在.

定理 2　中国剩余定理: 假设有 $n_1, n_2, \cdots n_k$ 个素数,在 0 到 $N(=n_1 * n_2 * \cdots n_k)$ 间存在一个同余值 SC,分别用 SC 对其素数求余.

$SC \bmod n_1 = a1$

$SC \bmod n_2 = a2$

…

$SC \bmod n_k = ak$

例如,假设素数 2、3、5 根据定理 2 在 0 ~ 30 之间存在一个同余值 $SC = 23$ 满足 $SC \bmod 2 = 1$, $SC \bmod 3 = 2$, $SC \bmod 5 = 3$.

素数编码是一种特殊的编码格式,用素数为结点编码. 根结点的素数编码(Prime_lable)为 2,自顶向下逐层为每个结点指定一个唯一的素数,即 self_lable,则每个结点的 Prime_lable 是其父结点 Prime_lable 和自己 self_lable 的乘积.

对于任意 2 个结点 u 和 v,若 v. Prime_lable 能被 u. Prime_lable 整除,则 u 是 v 的祖先结点. 若 v. Prime_lable 除以 u. Prime_lable 得到一个素数,则 u 是 v 的父结点.

供应链路径(path)由一系列的位置 L_i 按一定的顺序组成,如 $L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_n$ 就是一条路径,用树形结构来表示不同的移动路径,图 3 显示路径记录,图 4 显示路径记录的树形结构. 每个节点代表一个位置,用一个素数表示,灰色的节点表示产品的最终位置. 路径存储形式为($Element\_enc$, SC),其中 $Element\_enc$ 表示该路径上所有素数之积,SC 表示该路径的同余值,用于计算各结点间先后顺序.

path1:A→B→C
path2:A→B→D
path3:A→E→C
path4:A→D
path5:A→E

图 3　路径记录

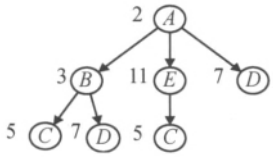

图 4　路径记录树

例如,已知 $Element\_enc = 30$, $SC = 23$,根据素数分解的唯一性,将 30 分解为 2 代表位置 A,3 代表位置 B 和 5 代表位置 C, $SC\%2 = 1$, $SC\%3 = 2$, $SC\%5 = 3$,所以该路径的先后顺序是 A→B→C.

考虑到路径可能重复,相同的素数应该返回不同的顺序无法实现,使用为相同的位置节点重命名的方式来解决这个问题.

例如 A→B→C→A→B→D→A,A 出现在第 1,4,7 的位置,重命名为 A_1, A_2, A_3,用类似的方法中命名该路径 $A_1 \rightarrow B_1 \rightarrow C_1 \rightarrow A_2 \rightarrow B_2 \rightarrow D_1 \rightarrow A_3$,由于相同的位置被认为是不同的节点可以计算它们的顺序编码.

考虑到路径过长,元素列表编码和顺序编码可能超出数据库存储范围,采用路径分裂技术将路径分成多个片段. 路径信息 E_ENC_1, …, E_ENC_m 为元素列表编码, O_ENC_1, …, O_ENC_m 为顺序编码. 如果路径片段数 t 小于 m,将以 1 填充元素编码 E_ENC_$(t+1)$, …, E_ENC_m,用 0 填充顺序编码 O_ENC_$(t+1)$, …, O_ENC_m.

如路径 $L_1 \rightarrow L_2 \rightarrow \cdots L_n$,被分为

$S_1 = L_1 \rightarrow L_2 \rightarrow \cdots L_{i1}$,

$S_2 = L_{i1}+1 \rightarrow L_{i1}+2 \rightarrow \cdots L_{i2}$,

…,

$S_{t-1} = L_{it-1}+1 \rightarrow L_{it-1}+2 \rightarrow \cdots L_{it}\ (1 \leqslant i_1 \leqslant i_2 \leqslant \cdots \leqslant i_t \leqslant n)$.

分裂的路径应该满足该条件:

$$\prod_{j=i_{u-1}+1}^{j=i_u} \mathrm{Prime}(L_i) \leqslant \mathrm{Max}_{\mathrm{Type}} \text{ and } \left(\prod_{j=i_{u-1}+1}^{j=i_u} \mathrm{Prime}(L_i)\right) \times \mathrm{Prime}(Li_u + 1) > \mathrm{Max}_{\mathrm{Type}}$$

$$u \in 1,2,3,\cdots,t \text{ and } i_0 = 0 \text{ and } \mathrm{Prime}(L_n + 1) = \infty .$$

Type: 元素列表编码和顺序编码必须是 RDBMS 支持的数据类型(如 BIGINT,DECIMAL 等).

$\mathrm{Max}_{\mathrm{Type}}$: 数据类型支持的最大值.

m: 分离路径的片段数.

例如: 假设路径 A→B→C→D→E→F,$P_1 = \mathrm{Prime}(\mathrm{A}) = 2$,$P_2 = \mathrm{Prime}(\mathrm{B}) = 3$,$P_3 = \mathrm{Prime}(\mathrm{C}) = 5$,$P_4 = \mathrm{Prime}(\mathrm{D}) = 7$,$P_5 = \mathrm{Prime}(\mathrm{E}) = 11$,$P_6 = \mathrm{Prime}(\mathrm{F}) = 13$ 和 $\mathrm{Max}_{\mathrm{Type}} = 250$. 由于 $\sum_{j=1}^{j=6} P_j = 30030 > \mathrm{Max}_{\mathrm{Type}} = 250$ 元素列表编码不能够存储,所以要分裂路径. 根据

$$\sum_{j=1}^{j=4} p_j = 210 < \mathrm{Max}_{\mathrm{Type}} \text{ and } \left(\sum_{j=1}^{j=4} p_j\right) \times p_5 = 2310 > \mathrm{Max}_{\mathrm{Type}} ,$$

$$\sum_{j=5}^{j=6} p_j = 143 < \mathrm{Max}_{\mathrm{Type}} \text{ and } \left(\sum_{j=5}^{j=6} p_j\right) \times p_7 = \infty > \mathrm{Max}_{\mathrm{Type}} .$$

路径被分为 S_1 = A→B→C→D ,S_2 = E→F.

2.2 存储策略

为了支持有效的处理物品的追踪查询和路径的查询,设计了一个存储策略如图 5 显示,关于路径记录编码后的数据会存储到表 PATH_TABLE 中,从该表中的数据可以恢复整个路径信息,关于产品详细信息数据存放到表 INFO_TABLE,IN_TABLE 表为中间表关联 PATH_TABLE 和 INFO_TABLE 表.

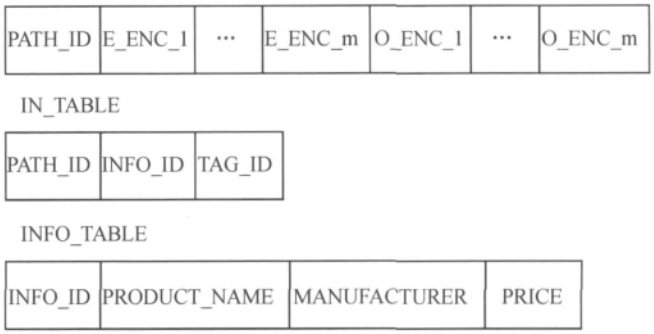

图 5 表结构

2.3 查询转换

定义 1 定义了几种查询模板(表 1).

表 1 查询模板

Query	Query	描述
Query1	<pathid = XXX>	查询 pathid 为 XXX 的移动路径
Query2	<Productname = 'Beef'>	查询牛肉的移动路径
Query3	<//L1//…//Ln>	查询经过 L1…,Ln 位置所以产品信息
Query4	<COUNT(),//L1//…//Ln >	查询经过 L1…,Ln 位置所以产品总数

定理 3 任何自然数 E_1,E_2 和 P 都满足 $(E_1 \times E_2) \bmod P = (E_1 \bmod P) \times (E_2 \bmod P) \bmod P$.

例如 $(3 \times 5) \bmod 2 = (3 \bmod 2) \times (5 \bmod 2)$.

证明 易证.

定理 4 E,P_1 和 P_2 是任意自然数,且 P_1 和 P_2 是素数则有,当且仅当 $E \bmod P_1 = 0$ 和 $E \bmod P_2 = 0$ 时,$E \bmod (P_1 \times P_2) = 0$.

例如 当 $30 \bmod 3 = 0$ 且 $3O \bmod 5 = 0$ 则 $30 \bmod (3 \times 5) = 0$.

证明 易证.

定理 5 O 为没分离原始路径顺序编码,O 为分离后第 i 段路径顺序编码 O_ENC_I,在路径中的每个位置 L_j 都满足 $\mathrm{O}\bmod P_j = \sum_{j=1}^{m} (O_j \bmod P_j) \left[1 - \frac{E_I \bmod P_j}{p_j}\right] = O_i \bmod P_j$.

证明 如果 L_j 被包含在第 i 段路径片段中,$E_i \bmod P_j = 0$,$\left[1 - \frac{E_I \bmod P_j}{p_j}\right] = 1$ 则 $(O_i \bmod P_j)$

$\left[1-\frac{E_i \bmod P_j}{p_j}\right]=O_i \bmod P_j$; 如果 L_j 不包含在第 i 段路径片段中,$E_i \bmod P_j \neq 0$,$\left[1-\frac{E_I \bmod P_j}{p_j}\right]=0$ 则($O_i \bmod P_j$) $\left[1-\frac{E_I \bmod P_j}{p_j}\right]=0$ 证得 $O \mod P_j = O_i \bmod P_j$ 即为该位置所在路径的顺序.

P_i 是 Prime(L_i) ,P 是 $P_1 \times P_2 \times \cdots P_k$,$E_i$ 是 E_ENC_i 时为了查询路径是否包含 L_1,L_2,…,L_k,因该检测该条件($E_1 \times E_2 \times \cdots \times E_i$) $\bmod P=0$ 是否成立,此时必须考虑是否溢出问题.

3　算法实现

图 6 显示存储路径编码的算法,算法的输入为路径记录,被转换为关系数据,先将相关数据存储在 txt 文件中,然后用块加载到 RDBMS 中. PATH_FILE 是 PATH_TABLE 的文件,TEMP_PATH_FILE 是 TEMP_PATH_TABLE 的文件.

```
Function Store_path_record (trace records tr)
Begin
1: //m:路径分离的片段最大数
2:m:=0
3:for i =0;i<the number of path records;i++
4:{
5:   <path_id, store_flag >:=constructTree(tree, tr[i])
6:   if store_flag==FALSE
7:    {
8:      <segment_list, segment_num>:=divie_path(tr[i])
9:      if segment_num>m
10:        m:=segment_num
11:     store segment_list for tr[i] in PATH_FILE
              using the path encoding scheme
12:   }
13: store(tag_id from tr[i],paht_id) in TEMP_PATH_FILE
14:}
15:create the schema according to m
16:perform bulk loading with PATHFILE, TEMP_PATH_FILE
17:after joining TEMP_PATH_TABLE on TAG_ID, fill IN_TABLE
18:end
```

图 6　路径编码算法

在第 2 行,初始化 m,然后根据路径记录用 constructTree(tree,tr) 构建路径树,如果在该树中 tr 路径不存在则插入一个新的路径并将 store_flag 设为 FALSE,否则返回 path_id 并设置 store_flag 为 TRUE.

在 6 ~ 12 行,如果 store_flag 为 false 时,编码不能存储,使用 divide_path(tr) 将路径分离成几个片段,返回路径片段列表和片段数目并保持最大的片段数目到 m,为每个路径片段进行编码存储; 否则直接将元素编码和顺序编码存入 PATH_FILE 文件.

在 13 行,保存 info_id 和 path_id 到 TEMP_PATH_FILE 文件中,用来填充 IN_TABLE 表.

```
Fouction element_translation(element list L1, L2,…, Lk in the path condition )
Begin
1;product:=1
2:condition:=null
//使用理论2将p分解
3:for i:=1;i<=k;i++
4:{
5: if(product*primre(Li))2> Max_Type
6: {
7:   if condition=null
8:     //m为最大片段数
9:     condition:=sub_element_translation(product,m)+" =0"
10:  else
11:  {
12:     condition:=condition+" and" +sub_element_translation(product,m)+" =0"
13:  }
14:  Product:=1
15: }
16: product:=product*pRimre(Li)
17:}
//p不需要使用理论2分解
19:If condition=null
20:   condition:=sub_element_translation(product,m)+" =0"
21:else
22:   condition:=condition+" and" +sub_element_translation(product,m)+" =0"
23:return condition
24:End

//使用理论1
Function sub_element_translation(product p,index i)
Begin
1:if i==1
2:    return  "E_ENC_" +i+" mod" +p
3:else
4:    return   "(" +sub_element_translation(p,i-1)+")(E_RNC_" +i+" mod" +P
                +" )mod" +p
End
```

图 7　查询路径转换算法

最后加载这些 txt 文件到对应的表中,再用 tag_id 连接 TEMP_PATH_TABLE 表来填充 IN_TABLE 表(图 7) . 此处不考虑 INFO_TABLE 表中详细信息. 当 $m=1$ 时,不用考虑溢出问题直接判断 $E_1 \bmod P=0$; 当 $m>1$ 时,要考虑溢出问题,由于任意自然数 $A \bmod P<P$,所以 $P^2 <= \mathrm{Max}_{\mathrm{Type}}$ 条件下($E_1 \times E_2$) $\bmod P=0$. 使用理论 3 转换为($E_1 \bmod P$) $\times$ ($E_2 \bmod P$) $\bmod P=0$ 不会溢出; 在 $P^2 > \mathrm{Max}_{\mathrm{Type}}$ 条件下,必须用定

理 4 将 P 分解成 $P = P_1 \times P_2 \times \cdots \times P_t$ 且 $P_i^2 <= \mathrm{Max}_{\mathrm{Type}}(i = 1, 2, \cdots, t)$ 再使用理论 3 转换.

例如,考虑 $m = 2$,P 被分解成 $p1$、$p2$,时$(E_1 \times E_2) \bmod P = 0$ 先用定理 4 转换为$(E_1 \times E_2) \bmod P_1 = 0$ and $(E_1 \times E_2) \bmod P_2 = 0$,再使用定理 3 转换为$(E_1 \bmod P_1) \times (E_2 \bmod P_1) \bmod P_1 = 0$ and $(E_1 \bmod P_2) \times (E_2 \bmod P_2) \bmod P_2 = 0$.

最后使用定理 5 确定位置的顺序.

4 实验结果

4.1 编码记录

IN_TABLE 表为中间关联表保存 PATH_ID、INFO_ID、TAG_ID 信息. 当 $m = 1$ 时,PATH_TABLE 表记录如表 2; 当 $m = 2$ 时,PATH_TABLE 表记录如表 3.

表 2 当 $m = 1$ 时 PATH_TABLE 表记录

PATH_ID	E_ENC_1	O_ENC_1
1	30	23
2	42	17
3	110	13
4	14	9

表 3 当 $m = 2$ 时 PATH_TABLE 表记录

PATH_ID	E_ENC_1	E_ENC_2	O_ENC_1	O_ENC_2
1	30	1	23	0
2	6	7	5	3
3	22	5	13	3
4	14	1	9	0

根据路径编码方法可知,编码之后的路径信息可以由 PATH_TABLE 表进行还原,属于无信息丢失的转换,路径编码实际上标识了具有相同移动路径的物品.

4.2 $m = 1$ 时的查询模板转换结果

当 $m = 1$ 时不需要考虑溢出问题,用 IN_TABLE 表关联 PATH_TABLE 和 INFO_TABLE 查询,路径 <//A//B//C> 查询采用定理 1、2 转换查询条件(图 8).

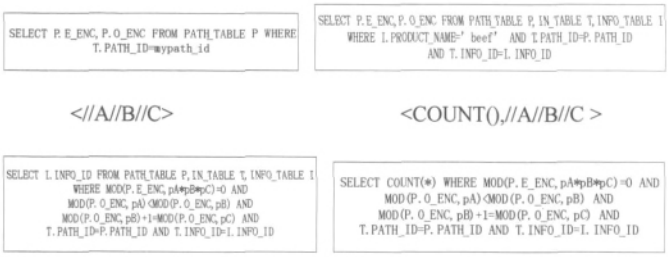

图 8 $m = 1$ 查询转换

4.3 $m > 1$ 时查询模板转换结果

图 9 显示了 $m = 2$ 时需要考虑溢出问题,用 IN_TABLE 表关联 PATH_TABLE 和 INFO_TABLE 查询,路径 <//A//B/C> 查询采用定理 1、2、3、4、5 转换查询条件,这里省略了其他的模板转换,除了路径转换外它们跟 $m = 1$ 是类似的.

```
SELECT I.INFO_ID FROM PATH_TABLE P, IN_TABLE T, INFO_TABLE I WHERE
MOD(MOD(P.E_ENC_1, pA*pB*pC)*MOD(P.E_ENC_2, pA*pB*pC), pA*pB*pC)=0 AND
MOD(P.O_ENC_1, pA)*FLOOR(1-(P.E_ENC_1 MOD pA)/pA)+(P.O_ENC_2, pA)*FLOOR
(1-(P.E_ENC_2 MOD pA)/pA)<MOD(P.O_ENC_1, pB)*FLOOR(1-(P.E_ENC_1 MOD pB)
/pB)+MOD(P.O_ENC_2, pB)*FLOOR(1-(P.E_ENC_2 MOD pB)/pB) AND
MOD(P.O_ENC_1, pB)*FLOOR(1-(P.E_ENC_1 MOD pB)/pB)+MOD(P.O_ENC_2, pB)*FLOOR
(1-(P.E_ENC_2 MOD pB)/pB)+1=MOD(P.O_ENC_1, pC)*FLOOR(1-(P.E_ENC_1 MOD pC)
/pC)+MOD(P.O_ENC_2, pC)*FLOOR(1-(P.E_ENC_2 MOD pC)/pC) AND
T.PATH_ID=P.PATH_ID AND T.INFO_ID=I.INFO_ID
```

图 9 $m = 2$ 查询转换

5 结　语

本文作者对目前常用的素数编码方法原理进行了分析,并指出它们在供应链环境中路径追溯应用中存在的优缺点. 素数编码查询性能较好且不存在更新问题,但其编码长度会随结点数量增加而导致溢出. 采用目前广泛使用的编码技术素数编码,对供应链中物品路径进行编码,主要解决编码过大溢出问题,设计了一个存储策略支持在关系型数据库上进行有效的查询处理. 最后提出了一种将查询模板转换为 SQL 查询的方法.

参考文献:

[1] 王伟. 浅析 RFID 技术对供应链管理的影响[J]. 商场现代化,2008(27):108.

[2] WANG F S,LIU P Y. Temporal management of RFID data [C]//Proceedings of the 31th International conference on Very Large Data Bases. San Francisco: Morgan Kaufmann,2005.

[3] LI Q Z,MOON B K. Indexing and Query XML Data for Regular Path Expression [C]//Proceeding of the 27th VLDB Conference. Roma: University of Arizona,2001:0.

[4] AMER-YAHIA S,FERNANDE X M,SFIVASTAVA D. Phase Matching in XML [C]//Proceeding of the 29th VLDB Conference. Berlin: Aalborg University,2003.

[5] COHEN E,KAPLAN H,MILO T. Labeling dynamic XMI tree [J]. SIAM Journal on Computing,2010,39(5):2048-2074.

[6] XU L,BAO Z F,LING T W. A dynamic Labeling Scheme Using Vectors [C]//Proceedings of the International Conference on Database and Expert Systems Applications. Berlin: Springer Berlin Heidelbers,2007.

[7] 孟小峰. XML 数据管理概念与技术[M]. 北京:清华大学出版社,2009.

[8] TATARINOV L,STRAIT D,BEGAT V K,et al. Storing and Querying Ordered XML Using a Relational Database System [C]//Proceedings of the 2002 ACM SIGMOD international conference on Management of data. New York: ACM,2002.

[9] WU X D,LEE M L,HSU W. A Prime Number Labeling Schemes for Dynamic Ordered XMLTrees [C]//Proceeding of the 20th Intemational Conferenee on Data Engineering ICDE,Singafore. University of Singafore,2004. 66-78.

[10] LEE C H,CHUNG C W. Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID [C]// ACM SIGMOD International Conference on Management of Data. Vancouver. BC: Acm Press,2008:291-302.

[11] LEE C H,CHUNG C W. RFID data processing in supply chain management using a path encoding scheme [J]. Knowledge and Data Engineerin,2011,23(5):742-758.

The research of the supply chain path coding based on prime encoding techonology

CHEN Xiong, WANG Xiao-mei

(College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China)

Abstract: Radio Frequency Identification (RFID) technology has been applied to many different fields. Especially, it is very useful in the monitoring and management of the supply chain. However, in such an environment, enormous RFID path information data will be generated, from which it therefore takes a long time to extract valuable information. In this paper, the path of items in the supply chain is effectively encoded by use of the prime number encoding which is currently a widely used coding technique and can not only compress the data volume but also is convenient to retrieve path information. A storage scheme is devised to support efficient query processing on an RDBMS . Finally, a method by which query templates are converted to SQL queries is proposed.

Key words: radio frequency identification; supply chain; prime coding; storage scheme; queries templates

(责任编辑: 包震宇)